

# Examen PP – Seria 2CC

29.05.2018

ATENȚIE: Aveți 2 ore · 100p pentru nota maximă · **Justificați** răspunsurile!

---

1. Reduceți la forma normală următoarea expresie, ilustrând pașii de reducere:  
 $\lambda x.\lambda y.((\lambda x.\lambda y.y (x y)) (y x))$  10p
2. Care este diferența între următoarele două linii de cod Racket  
`(let* ((a 1) (b 2) (c (+ a 2))) (+ a b c))`  
`((lambda (a b c) (+ a b c)) 1 2 (+ a 2))` 10p
3. Scrieți în Racket o funcție echivalentă cu `unzip` din Haskell, știind că  
`unzip :: [(a, b)] -> ([a], [b])`. Folosiți cel puțin o funcțională. 10p
4. Sintetizați tipul funcției `f` în Haskell: `f x y z = x . y z` 10p
5. Instanțiați clasa `Show` pentru funcții Haskell care iau un argument numeric, astfel încât afișarea unei funcții `f` va produce afișarea rezultatelor aplicării funcției pe numerele de la 1 la 10. E.g. afișarea lui `(+1)` va produce: `234567891011`. 10p
6. Folosiți list comprehensions pentru a produce fluxul listelor de divizori pentru numerele naturale: `[[1], [1, 2], [1, 3], [1, 2, 4], [1, 5], [1, 2, 3, 6] ...]`. 10p
7. Folosiți rezoluția pentru a demonstra că dacă *George este țaran* și *orice țaran are o sapă* atunci este adevărat că *George este deștept sau George are o sapă* (folosiți predicatul `țaran(X)`, `areSapă(X)` și `deștept(X)`). 10p
8. Scrieți un predicat Prolog `intersect(A, B, R)` care leagă `R` la intersecția mulțimilor (reprezentate ca liste) `A` și `B`.
9. Dat fiind un șir de date binare, scrieți un algoritm Markov care plasează la sfârșitul șirului suma modulo 2 a biților din șir. Exemple: `101010110000111`  
`→ 1010101100001110; 100110110110 → 1001101101101; 100110110111 → 1001101101110` 10p
10. Considerăm o structură de date de tip listă circulară, caracterizată de conținutul său și de un cursor intrinsec structurii, poziționat la orice moment pe un element al listei. Avem următoarele funcționalități:
  - Structura va putea fi creată pe baza unei liste obișnuite `L`; la crearea cursorul va fi inițial poziționat pe elementul care era ultimul element din `L`;
  - Operația `get`, care întoarce elementul de la poziția unde este cursorul;
  - Operația `prev`, care deplasează cursorul cu o poziție spre stânga;Exemplu: avem lista circulară `C`, construită pe baza listei `1,2,3,1,5`. Astfel:  
`get(C) = 5` `get(prev(prev(prev(prev(C)))))) = 1`  
`get(prev(C)) = 1` `get(prev(prev(prev(prev(prev(C)))))) = 5`  
Se cere implementarea în Racket, Haskell sau Prolog a celor 3 funcționalități: crearea listei circulare, operația `get` și operația `prev`. 10p fiecare funcționalitate